

max\_Synchronization.py and log\_Synchronization.py

1. Create three initial conditions  $i-j$ ,  $j-i$ , and  $\text{abs}(i-j)$  where  $(i,j)$  is a coordinate as the initial sample size.
2. Perform LPP or DPPE from start (line) to end (n) traveling as many steps as needed.
3. Plot the saved steps using Matplotlib and create a video.

cupySubprocesses.py and numpyNumbaSubprocesses.py

1. Choose a  $t_{\text{start}}$  and  $t_{\text{end}}$  along with how many increments you want to save from  $t_{\text{end}}$ . You can also set the initial condition.
2. Filenum will create as many processes you want so that you can run multiple processes to generate as many samples as you want.  $n_{\text{samples}}$  is how many samples each process will create.
3. It is recommended you choose 1 GPU and CPU Cores equal to the number of processes you want as it was found to be the most efficient.

pathCode.py

1. Performs regular LPP or DPPE, but also finds the geodesics of each point at each time step.
2. The cords function was used to find the intersection points of the coordinates  $(0,1)$  and  $(1,0)$  along with  $(1,-1)$  and  $(-1,1)$  using argmax from numpy.